**CSCI-E97 Software Design: Principles, Models, and Patterns**

**Assignment 4 - Smart City Authentication Service**

# Results Document

Date: 11/16/2020

Author: Samer Akash

o **Did the application of the design patterns help or hinder your design and implementation? Please explain how.**

I think the singleton and composite patterns helped a lot, but the visitor pattern was not easy to implement. The singleton pattern and composite pattern were perfect fit for my implementation. I felt Visitor pattern is little forced to fit within my implementation. I tried my best to stay within the spirit of Visitor pattern.

○ **How could the design have been better, clearer, or made the implementation easier?**

I think the design could be better if I was able to see the requirements for the rest the system components at once to have an idea of the big picture.

I will also think about redesigning the mapping between the restricted methods in the model service and required permission. Required permissions for restricted methods are hard coded now inside each method.

**○ Any implementation changes that you made to your design and how they continue to support the requirements?**

- Added two implementations for the Visitor interface, one for check access and oen for the inventory.

- Represented credentials in their own classes.

- to simplify the demo of command line processor, authToken value will be the same as the user id

- AuthTokens will timeout after 1 hour. it will also expire after each use in order to demo the command line processor. you will notice I login to get a new authToken before running any method.

- The script changed little bit to match to the rest of the assignment’s scripts.

- permission and roles command lines must run first to establish the controller service permissions.

- controller credentials are hardcoded and initiated in the controller constructor.

- new residents get default faceprint credentials at the creation in the model services.

- Required permission and roles are hardcoded in the model service public methods.

- passwords hashing function has a hardcoded key of "passwordpassword"

* **Is the design process getting easier?**

Yes. The more I learn about design patterns and after implementing some prototype code, you can start connecting the dots together. The review process also is getting smoother and easier

**○ Did the design review help improve your design?**

The reviewers helped me to identify couple of missed requirements and pointed out some changes needed regarding the class diagram. Also, after reviewing their designs I have noticed some missing items from my design doc.

**○ Comments about your design from your peer design review partners?**

**From Brett:**

I looked over your design and everything looks good. Good job especially on the sequence diagram.  Those can be pretty confusing to read at times but your sequence diagram was very intuitive, i could pretty much related everything in the diagram to what the application is supposed to be doing. I did have one question though about the class diagram. I'm curious as to what the purpose is of the InventoryAdd and InventoryUpdate concrete visitors? I understand the case for the CheckAccess visitor, but why not have a hash map that stores these permission, roles etc... objects in the Authentication Service then use put and update to manage those items?

My design right now uses hash maps for this, but I'm curious if maybe there was a better implementation i may have to looked over.

**From Nitya:**

Nice work! It seems like you're right on track so far. Please see my comments throughout the document below.

It is not clear what InventoryAdd and InventoryUpdate (in Visitor) refer to.

Very clear layout. Great structure!

It could be helpful to add some exceptions to the diagram above, in order to address scenarios where access is denied, the authtoken is invalid, or there is another problem with authentication.

You've probably already accounted for this, but just a quick reminder that Authentication Service methods are considered restricted (except for login and checkAccess)

Also, it could be helpful to color code different types of classes in the UML diagram so that it is easy to differentiate between different classes. Adjusting the structure could also help enhance readability. I believe the general guideline is to position inheritance relations top-down and composition left to right.

How are new visitor instances created? It could be good to mention how the Visitor pattern is demonstrated in this design

Great diagram! In the description, it could be helpful to explain the difference between authtoken1 vs authtoken2 vs authtoken 3.

It could be a good idea to note which aspects of the model service were modified during this process and perhaps even consider explaining why this was done.

It could be a good idea to include specific security risks here (DoS, man in the middle, traffic/activity monitoring, etc.)

**○ Comments provided by you for each of your peer design review partners**

**To Brett**:

- I'm not seeing in your class digram any modeling for Credentials. User should be associated to multiple credential types at the same time.  
- I think the Entitlement class should be Abstract class or an interface in order to comply with the composite pattern. It will make the implementation easier for you.  
- I'm not seeing an implementation for the visitor interface that manages the inventory. Based on the requirements and piazza discussions, i think we need to have at least two implementations for the visitor interface. One for traversing the inventory and one for the check access.  
- I don't think you need to associate between AuthToken class and the service implementation class. Structurally, the AuthToekn should belong to a single user not to the inventory.  
- You need a function to hash passwords before storing them. I will recommend adding that as private method in the service interface implementation class.  
- I think you will need to expose more public functions from the Authentication Service for: getting the singleton instance, adding resources, add resources to resource role. Im sure you will find more while implementation.  
- Im not seeing a sequence diagram in your design doc. Make sure to add it your design as it worth 4 points in the grading sheet.  
- I will recommend renaming AuthServiceVisitorElement to Visitable. The graders pointed out last lecture to follow the standard naming conventions because they get confused while grading.

**To Nitya:**

- The system architecture document mentioned a requirement to build all services as interfaces, this means that all public methods can be accessed only through this interface. Im seeing  
some public methods coming from your domain classes (Like User class) which means these methods are exposed outside of the interface.  
- We have another requirement to implement singleton pattern to get a single instance of the authentication service. I'm not seeing the modeling for this in the class diagram.  You will need getInstance() method to in your authenticating interface.  
- In the class diagram, If the Authenticator is your interfaces concrete class, then you can just add an interface and link it to it to show the instance of the authenticating interface.  
- In order to implement the visitor pattern based on the requirement, we need different implementations for the visitor interface. One implementation to check access and one for traversing the inventory. Im not able to identify these implementations from you class diagram.  I will recommend checking piazza, there are some questions about this.  
- I think you forgot to update the use cases diagram. I'm seeing only the use cases diagram of the controller service.  
- Sequence diagram is missing from your document. I will recommend building this diagram before starting the implementation. it will make your implementation easier.